Boosting & Randomized Forests for Visual Recognition

Jamie Shotton  Tae-Kyun Kim  Björn Stenger

Microsoft Research Cambridge  UNIVERSITY OF CAMBRIDGE  TOSHIBA

ICCV 2009, Kyoto, Japan

http://mi.eng.cam.ac.uk/~tkk22/iccv09_tutorial
Course Overview

Part I: Random Forests
   Jamie Shotton

Part II: Boosting
   Tae-Kyun Kim

Part III: Online Learning
   Björn Stenger

- Coffee break
  - half way through Part II

- Questions
  - please ask as we go

- References & web resources
  - at end of each part

- Notation
  - may differ slightly between parts

http://mi.eng.cam.ac.uk/~tkk22/iccv09_tutorial
Part I
Randomized Decision Forests

- Very fast tools for
  - classification
  - clustering
  - regression

- Good generalization through randomized training

- Inherently multi-class
  - automatic feature sharing [Torralba et al. 07]

- Simple training / testing algorithms

“Randomized Decision Forests” = “Randomized Forests” = “Random Forests™”
Randomized Forests in Vision

[Amit & Geman, 97] digit recognition

[Lepetit et al., 06] keypoint recognition

[Moosmann et al., 06] visual word clustering

[Shotton et al., 08] object segmentation

[Rogez et al., 08] pose estimation

[Criminisi et al., 09] organ detection

(Among many others...)
Outline

• Randomized Forests
  – motivation
  – training & testing
  – implementation
  – regression, clustering, max-margin, boosting

• Applications to Vision
  – keypoint recognition
  – object segmentation
  – human pose estimation
  – organ detection
The Basics: **Is The Grass Wet?**

- Is the world state wet?
  - $P(wet) = 0.95$
  - $P(wet) = 0.9$
  - $P(wet) = 0.1$

- Is it raining?
  - *no* $P(wet) = 0.1$
  - *yes* $P(wet) = 0.95$

- Is the sprinkler on?
  - *no* $P(wet) = 0.1$
  - *yes* $P(wet) = 0.9$
The Basics: Binary Decision Trees

- feature vector $\mathbf{v} \in \mathbb{R}^N$
- split functions $f_n(\mathbf{v}) : \mathbb{R}^N \to \mathbb{R}$
- thresholds $t_n \in \mathbb{R}$
- classifications $P_n(c)$

Diagram:

- Node 1: $f_1(\mathbf{v}) \leq t_1$
- Nodes 2 and 3:
  - Node 2: $f_3(\mathbf{v}) \leq t_3$
  - Node 3: $f_6(\mathbf{v}) \leq t_6$
- Node 4:
  - Node 5: $f_10(\mathbf{v}) \leq t_{10}$
- Nodes 6 and 10:
  - Node 6: $f_10(\mathbf{v}) \leq t_{10}$
  - Node 10: $P_{17}(c)$
- Nodes 7, 11, 12, and 13:
  - Node 11:
    - Node 12:
      - Node 13:
  - Node 12:
    - Node 13:
  - Node 11:
    - Node 12:
      - Node 13:

Legend:
- Leaf nodes
- Split nodes

Category C
double[] ClassifyDT(node, v)
    if node.IsSplitNode then
        if node.f(v) >= node.t then
            return ClassifyDT(node.right, v)
        else
            return ClassifyDT(node.left, v)
        end
    else
        return node.P
    end
end
Toy Learning Example

- Try several lines, chosen at random
- Keep line that best separates data
  - information gain
- Recurse

- feature vectors are $x, y$ coordinates: $v = [x, y]^T$
- split functions are lines with parameters $a, b$: $f_n(v) = ax + by$
- threshold determines intercepts: $t_n$
- four classes: purple, blue, red, green
Toy Learning Example

- Try several lines, chosen at random
- Keep line that best separates data
  - information gain
- Recurse

- feature vectors are $x, y$ coordinates: $v = [x, y]^T$
- split functions are lines with parameters $a, b$: $f_n(v) = ax + by$
- threshold determines intercepts: $t_n$
- four classes: purple, blue, red, green
Toy Learning Example

- Try several lines, chosen at random
  - feature vectors are $x$, $y$ coordinates: $\mathbf{v} = [x, y]^T$
  - split functions are lines with parameters $a$, $b$: $f_n(\mathbf{v}) = ax + by$
  - threshold determines intercepts: $t_n$
  - four classes: purple, blue, red, green

- Keep line that best separates data
  - information gain

- Recurse
Toy Learning Example

- Try several lines, chosen at random
- Keep line that best separates data
  - information gain
- Recurse

- feature vectors are $x, y$ coordinates: $\mathbf{v} = [x, y]^T$
- split functions are lines with parameters $a, b$: $f_n(\mathbf{v}) = ax + by$
- threshold determines intercepts: $t_n$
- four classes: purple, blue, red, green
Randomized Learning

• Recursively split examples at node \( n \)
  – set \( I_n \) indexes labeled training examples \((v_i, l_i)\):

\[
\begin{align*}
I_1 &= \{ i \in I_n \mid f(v_i) < t \} \\
I_r &= I_n \setminus I_1
\end{align*}
\]

• At node \( n \), \( P_n(c) \) is histogram of example labels \( l_i \)
More Randomized Learning

- **Features** $f(v)$ chosen at random from feature pool $f \in \mathcal{F}$

- **Thresholds** $t$ chosen in range $t \in (\min_i f(v_i), \max_i f(v_i))$

- **Choose** $f$ and $t$ to maximize gain in information

$$\Delta E = -\frac{|I_1|}{|I_n|} E(I_1) - \frac{|I_r|}{|I_n|} E(I_r)$$

Entropy $E$ calculated from histogram of labels in $I$
Implementation Details

• How many features and thresholds to try?
  – just one = “extremely randomized”  [Geurts et al. 06]
  – few -> fast training, may under-fit, maybe too deep
  – many -> slower training, may over-fit

• When to stop growing the tree?
  – maximum depth
  – minimum entropy gain
  – delta class distribution
  – pruning
Randomized Learning Pseudo Code

TreeNode LearnDT(I)

repeat featureTests times
  let f = RndFeature()
  let r = EvaluateFeatureResponses(I, f)

repeat threshTests times
  let t = RndThreshold(r)
  let (I_l, I_r) = Split(I, r, t)
  let gain = InfoGain(I_l, I_r)
  if gain is best then remember f, t, I_l, I_r
end

if best gain is sufficient
  return SplitNode(f, t, LearnDT(I_l), LearnDT(I_r))
else
  return LeafNode(HistogramExamples(I))
end
Training Strategies

**depth first**

- **Recursive algorithm**
  - partitions all training examples

- **Store all images in memory**
  - can be memory hungry

**Good for**
- smaller data sets
- deeper trees

**breadth first**

- **One pass through data per tree level**
  - can load images on-the-fly

- **Maintain 4D histogram of size**
  \[2^d \times F \times T \times C\]
  - no. nodes
  - no. features
  - no. thresholds
  - no. classes

**Good for**
- very large data sets
- shallower trees
GPU Acceleration

- **GPUs can dramatically accelerate**
  - training – 10x speed-up – breadth first
  - testing – 100x speed-up

- **Tree is encoded as GPU texture**

- **Caveats**
  - some limitations on image features
  - implementation requires considerable GPU know-how

[Sharp 08]
Binary Decision Trees Summary

• Fast greedy training algorithms
  – can search infinite pool of features
  – heterogeneous pool of features

• Fast testing algorithm

• Needs careful choice of hyper-parameters
  – maximum depth
  – number of features and thresholds to try

• Prone to over-fitting
A Forest of Trees

• Forest is ensemble of several decision trees

\[ P(c|\mathbf{v}) = \frac{1}{T} \sum_{t=1}^{T} P_t(c|\mathbf{v}) \]

[Amit & Geman 97]
[Breiman 01]
[Lepetit et al. 06]
double[] ClassifyDF(forest, v)
  // allocate memory
  let P = double[forest.CountClasses]

  // loop over trees in forest
  for t = 1 to forest.CountTrees
    let P' = ClassifyDT(forest.Tree[t], v)
    P = P + P' // sum distributions
  end

  // normalise
  P = P / forest.CountTrees
end
Learning a Forest

• Divide training examples into $T$ subsets $I_t \cup I$
  – improves generalization
  – reduces memory requirements & training time

• Train each decision tree $t$ on subset $I_t$
  – same decision tree learning as before

• Multi-core friendly

• Subsets can be chosen at random or hand-picked
• Subsets can have overlap (and usually do)
• Can enforce subsets of images (not just examples)
• Could also divide the feature pool into subsets
Learning a Forest **Pseudo Code**

```plaintext
Forest LearnDF(countTrees, I)
    // allocate memory
    let forest = Forest(countTrees)

    // loop over trees in forest
    for t = 1 to countTrees
        let I_t = RandomSplit(I)
        forest[t] = LearnDT(I_t)
    end

    // return forest object
    return forest
end
```
Toy Forest Classification Demo

6 classes in a 2 dimensional feature space. Split functions are lines in this space.
With a depth 2 tree, you cannot separate all six classes.
With a depth 3 tree, you are doing better, but still cannot separate all six classes.
With a depth 4 tree, you now have at least as many leaf nodes as classes, and so are able to classify most examples correctly.
Different trees within a forest can give rise to very different decision boundaries, none of which is particularly good on its own.
Toy Forest Classification Demo

But averaging together many trees in a forest can result in decision boundaries that look very sensible, and are even quite close to the max margin classifier. (Shading represents entropy – darker is higher entropy).
Tree outputs and objective functions

- **Trees can be trained for**
  - classification, regression, or clustering

- **Change the object function**
  - information gain for classification: 
    \[ I = H(S) - \sum_{i=1}^{2} \frac{|S_i|}{|S|} H(S_i) \]
    measure of distribution purity
Regression trees

- Real-valued output $y$
- Object function: maximize $\text{Err}(S) - \sum_{i=1}^{2} \frac{|S_i|}{|S|} \text{Err}(S_i)$

measure of fit of model

$\text{Err}(S) = \sum_{j \in S} (y_j - y(x_j))^2$

e.g. linear model $y = ax + b$, Or just constant model
Clustering trees

- Output is cluster membership

- Option 1 – minimize imbalance:
  \[ B = |\log|S_1| - \log|S_2|| \]  
  [Moosmann et al. 06]

- Option 2 – maximize Gaussian likelihood:
  \[ T = |\bigwedge S| - \sum_{i=1}^{2} \frac{|S_i|}{|S|}|\bigwedge S_i| \]  
  measure of cluster tightness (maximizing a function of info gain for Gaussian distributions)
Clustering example

**Visual words good for e.g. matching, recognition but** $k$-means clustering **very slow**

**Randomized forests for clustering descriptors**
- e.g. SIFT, texton filter-banks, etc.

**Leaf nodes in forest are clusters**
- concatenate histograms from trees in forest

[Moosmann et al. 06]

[Sivic et al. 03]
[Csurka et al. 04]
Clustering example

[Moosmann et al. 06]

```
\begin{itemize}
  \item tree $t_1$
  \item tree $t_T$
\end{itemize}

\textbf{“bag of words”}

\begin{itemize}
  \item \textbf{node index}
  \item \textbf{frequency}
\end{itemize}

```

frequency

node index
Relation to other parts of this tutorial

- **Boosting (Part II)**
  - decision trees as weak learners
  - boosted classifiers as split functions [Tu 05]

- **Online learning (Part III)**
  - trees can be updated ‘online’ [Yeh et al. 07]
    - distributions of leaves
    - structure of tree
Relation to Cascades

- **Boosted Cascades**
  - very unbalanced tree
  - good for unbalanced binary problems
    e.g. sliding window object detection

- **Randomized forests**
  - less deep, fairly balanced
  - ensemble of trees gives robustness
  - good for multi-class problems
Relation to Max-Margin Classifiers

- Max-margin split functions
  - split functions have built-in generalization

- Tree of max-margin classifiers (SVMs)
  - recursively partition set of classes down the tree
Random Ferns

- Naïve Bayes classifier over random sets of features

\[ P(C|f_1, \ldots f_N) \propto P(f_1, \ldots f_N|C)P(C) \]

Bayes’ rule

\[ \approx \prod_{j=1}^{N} P(f_j|C) \]

“ naïve Bayes”

individual features

\[ \approx \prod_{k=1}^{M} P(F_k|C) \]

“random ferns”

set of features

- Can be good alternative to randomized forests

[Özuysal et al. 07]
[Bosch et al. 07]
Outline

• Randomized Forests
  – motivation
  – training & testing
  – implementation
  – regression, clustering, max-margin, boosting

• Applications to Vision
  – keypoint recognition
  – object segmentation
  – human pose estimation
  – organ detection
Fast Keypoint Recognition

- Wide-baseline matching as classification problem
- Extract prominent key-points in training images
- Forest classifies
  - patches -> keypoints
- Features
  - pixel comparisons
- Augmented training set
  - gives robustness to patch scaling, translation, rotation
Fast Keypoint Recognition

[Lepetit et al. 06]

• Example videos
Real-Time Object Segmentation [Shotton et al. 2008]

- Segment image and label segments in real-time

CVPR 2008 Best Demo Award!
Object Recognition Pipeline

- **extract features**
  - SIFT, filter bank

- **clustering**
  - k-means

- **assignment**
  - nearest neighbour

**classification algorithm**

- SVM, decision forest, boosting

**hand-crafted**

**unsupervised**

**supervised**
Object Recognition Pipeline

Semantic Texton Forest (STF)
- decision forest for clustering & classification
- tree nodes have learned object category associations

classification algorithm
SVM, decision forest, boosting
Example Semantic Texton Forest

Input Image


A[b] + B[b] > 284

A[g] - B[b] > 28

A[b] > 98

|A[b] - B[g]| > 37


|A[b] - B[g]| > 37

A[b] > 98


A[b] + B[b] > 284

A[g] - B[b] > 13

P(c|l)

Example Patches
Leaf Node Visualization

- Average of all training patches at each leaf node
Semantic Textons & Local Classification

test image

semantic textons
(color ↔ leaf node index)

local classification
(color ↔ most likely category)

ground truth
(for reference)

comparable
Segmentation Forest

- Object segmentation

- Adapt TextonBoost [Shotton et al. 07]
  - boosted classifier → randomized decision forest
textons
generalized
semantic textons
MSRC Dataset Results
3D Point-Cloud Features

- [Brostow et al. 08]
  - structure-from-motion cues for object segmentation
Human Pose Estimation  [Rogez et al. 08]

- Torus defined on
  - dimension 1: cyclical action (e.g. walking)
  - dimension 2: camera view point (360 degrees)

- Discrete bins on the torus used as classes in random forest
Organ Recognition

[Criminisi et al. 09]

- Quickly localize bodily organs in 3D CT scans

Decision Forests with Long-Range Spatial Context for Organ Localization in CT Volumes

A. Criminisi, J. Shotton and S. Bucciarelli
Microsoft Research Ltd. Cambridge, UK

Brain Segmentation

[Yi et al. MICCAI 09]

ground truth

result

<table>
<thead>
<tr>
<th>Method</th>
<th>CSF</th>
<th>GM</th>
<th>WM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive MAP</td>
<td>0.069</td>
<td>0.564</td>
<td>0.567</td>
</tr>
<tr>
<td>Biased MAP</td>
<td>0.071</td>
<td>0.558</td>
<td>0.562</td>
</tr>
<tr>
<td>Fuzzy c-means</td>
<td>0.048</td>
<td>0.473</td>
<td>0.567</td>
</tr>
<tr>
<td>Maximum-a-posteriori (MAP)</td>
<td>0.071</td>
<td>0.550</td>
<td>0.554</td>
</tr>
<tr>
<td>Maximum-likelihood</td>
<td>0.062</td>
<td>0.535</td>
<td>0.551</td>
</tr>
<tr>
<td>Tree-Structure k-means</td>
<td>0.049</td>
<td>0.477</td>
<td>0.571</td>
</tr>
<tr>
<td>MPM-MAP [11]</td>
<td>0.227</td>
<td>0.662</td>
<td>0.683</td>
</tr>
<tr>
<td>MAP with histograms</td>
<td>0.549 ± 0.017</td>
<td>0.814 ± 0.004</td>
<td>0.710 ± 0.005</td>
</tr>
<tr>
<td>Decision Forest Classifier</td>
<td>0.614 ± 0.015</td>
<td>0.838 ± 0.006</td>
<td>0.731 ± 0.007</td>
</tr>
</tbody>
</table>
Take Home Message from Part I

• Randomized decision forests
  – very fast

  – accuracy comparable with other classifiers

  – simple to implement

  – extremely flexible tools for computer vision
References (red = most relevant)

- Amit & Geman
  - Shape Quantization and Recognition with Randomized Trees.
- Besl et al.
  - Image Classification using Random Forests and Ferns.
- Brox et al.
  - Random Forests.
- Breiman et al.
  - Classification and Regression Trees
- Breitenstein et al.
  - Segmentation and Recognition using Structure from Motion Point Clouds.
  - ECV 2006.
- Csurka et al.
  - Visual Categorization with Bags of Keypoints.
- Fuchs & Buhmann
  - Inter-Active Learning of Randomized Tree Ensembles for Object Detection.
- Geurts et al.
  - Extremely Randomized Trees.
- Grauman & Darrell
  - The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features.
  - ICCV 2005.
- Hua et al.
  - Discriminant Embedding for Local Image Descriptors.
- Jurie & Triggs
  - Creating Efficient Codebooks for Visual Recognition.
  - CVPR 2006.
- Leibe et al.
  - Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories.
  - CVPR 2006.
- Leptitch et al.
  - Keypoint Recognition using Randomized Trees.
  - PAMI 2005.
- Lowe
  - Distinctive image features from scale-invariant keypoints.
- Malik et al.
  - Contour and Texture Analysis for Image Segmentation.
- Mikolajczyk & Schmid
  - Scale and Affine Invariant interest point detectors.
- Mosmann et al.
  - Fast Discriminative Visual Codebooks using Randomized Clustering Forests.
  - NIPS 2006.
- Nister & Stewenius
  - Scalable Recognition with a Vocabulary Tree.
  - CVPR 2006.
- Ouyang et al.
- Rogez et al.
  - Randomized Trees for Human Pose Detection.
  - CVPR 2008.
- Sharp
  - Implementing Decision Trees and Forests on a GPU.
  - ECCV 2008.
- Shotton et al.
  - Semantic Texton Forests for Image Categorization and Segmentation.
  - CVPR 2008.
- Shotton et al.
- Sivic & Zisserman
  - ICCV 2003.
- Tieleman & Hinton
  - Margin trees for high-dimensional classification.
- Torralba et al.
  - Sharing visual features for multiclass and multiview object detection.
  - Tu
  - Probabilistic Boosting-Tree: Learning Discriminative Models for Classification, Recognition, and Clustering.
  - ICCV 2005.
  - Tu
  - Auto-context and its application to high-level vision tasks.
  - CVPR 2008.
  - Tuytelaars & Schmid
  - Vector Quantizing Feature Space with a Regular Lattice.
  - Verbeek & Schmid
  - A statistical approach to texture classification from single images.
  - PAMI 2005.
- Verbeek & Triggs
  - Region-Classification with Markov Field Aspect Models.
- Viola & Jones
  - Robust Real-Time Object Detection.
  - UC 2004.
- Winn et al.
  - Object Categorization by Learned Universal Visual Dictionary.
  - ICCV 2005.
- Wu et al.
  - Enlarging the Margins in Perceptron Decision Trees.
- Yeh et al.
Web Resources on Random Forests

• Tutorial Webpage
  - http://mi.eng.cam.ac.uk/~tkk22/iccv09_tutorial

• Leo Breiman’s Webpage
  - http://www.stat.berkeley.edu/~breiman/RandomForests

• Regression Trees
End of Part I

Thank You

jamie@shotton.org

Internships at Microsoft Research Cambridge available for next spring/summer. Talk to me or see:

http://research.microsoft.com/en-us/jobs/intern/about_uk.aspx